REMARKS

The Office Action of June 23, 2004 has been received and its contents carefully noted.

In response to the rejections for indefiniteness, in sections 2-9 of the Office Action, the present Amendment changes "pre-ICB program position" in the independent claims to "pre-initial conditional breakpoint program position." Accordingly, the rejections for indefiniteness should be withdrawn.

In response to the rejection of claims 1-11 and 34-44 in section 11 of the Office Action, the present Amendment revises the preambles of independent claims 1 and 34 to specify a "computer-readable medium encoded with a program debugger ...", and adjusts the preambles of the relevant dependent claims to conform to the new preambles of the independent claims. Accordingly, the rejections for non-statutory subject matter should be withdrawn.

Section 13 of the Office Action rejects all of the independent claims (and some of the dependent claims) for anticipation by an article by Robert Wahbe et al entitled "Practical Data Breakpoints: Design and Implementation." Section 15 of the Office Action rejects the remaining claims for obviousness on the basis of the same reference. For the reasons discussed below, however, it is respectfully submitted that the independent claims are patentable over Wahbe et al (which will hereafter be called simply "Wahbe").

The Wahbe article is directed to a way to detect what the reference calls "data breakpoints." The reference explains, "*Data breakpoints* specify the break condition in terms of the program's memory state, for example stop when playing field f of structure s is

modified," so that a debugging task such as "print the value of field f of structure s every time it is updated" can be performed (page 1 of Wahbe, middle of column 2). This differs from what Wahbe calls a "control breakpoint," which specifies the break condition in terms of the program's control flow.

Wahbe's Abstract states, "A data breakpoint facilities must monitor all memory updates performed by the program being debugged." But actually checking every memory update would impose high computational overhead, so Wahbe employs a bitmapping scheme "for checking whether an individual target address is part of the data breakpoint condition" (page 2, middle of column 2). Otherwise, it can be ignored. Wahbe summarizes a second stage in reducing the computational overhead as follows (page 2, bottom portion of column 2):

> Second, we investigate the additional performance benefit of eliminating write checks through compile-time analysis. Our strategy for eliminating write checks has three components. First, we use a symbol table matching algorithm to find as many known write instructions as possible. We check those instructions only when the variables to which they write occur in break conditions. Second, we use data flow analysis techniques to remove checks on write instructions within loops. We replace such checks by checks that execute only once, on entry to the loop. Third, we support these loop optimizations by using data structures that provide efficient checks on contiguous *ranges* of memory locations. We demonstrate that, at the cost of considerable implementation complexity, these techniques dramatically diminish the dynamic count of checked write instructions.

Later, in a "Loop Optimization" section in the second column of page 8, the reference explains that all loop invariant target addresses are detected, and the checks for

these loop invariant addresses are eliminated and replaced "with write checks in a pre-header block that dominates all entrances to the loop."

It is respectfully submitted that an ordinarily skilled person would understand that Wahbe's approach to determining data breakpoints essentially seeks to efficiently find instructions that modify memory locations storing features of interest. When he detects write instructions that might cause problems, he checks them, and if such write instructions occur in the loop he conducts the checks at the entrance of the loop instead of every iteration though the loop. This is quite different from the debugging technique disclosed in the present application.

Turning now to claim 1, the Office Action (on page 5) identifies several passages in the Wahbe article as disclosing the first three "means" clauses of the claim. There is a fourth "means" clause in claim 1, though: "means for reestablishing said initial conditional breakpoint if said special conditional breakpoint is satisfied." An ordinarily skilled person would likely think that Wahbe teaches finding write instructions that should be checked, and if such a write instruction occurs in a loop, it can be checked outside the loop instead of repeatedly as the loop is executed. No means is disclosed in the reference "for reestablishing said initial conditional breakpoint if said special conditional breakpoint is satisfied." Nor would an ordinarily skilled person who was guided by the reference modify what the reference discloses so as to include the fourth "means" of claim 1.

Contrary to the position taken in the Office Action, it is also respectfully submitted that the reference does not really disclose the first "means" of claim 1: "means for extracting, from an initial conditional breakpoint within a program loop, a first Boolean

expression that is at least partially invariant within the loop." The sentence in the second column on page 8 of the reference that is identified in the Office Action just says that all loop invariant target addresses are detected, not that a Boolean expression is extracted from a conditional breakpoint in a loop.

Independent claim 12 recites "extracting, from an initial conditional breakpoint within a program loop, a first Boolean expression that is at least partially invariant within the loop," and "if said special conditional breakpoint is satisfied, reestablishing said initial conditional breakpoint." Independent claim 23 includes similar limitations. It is respectfully submitted that claims 12 and 23 are patentable over the reference for reasons along the lines discussed above with respect to claim 1.

Independent claim 34 recites "means for extracting, from an initial conditional breakpoint within a program loop, a first Boolean expression that is at least partially invariant within the loop." Independent claim 45 has a similar "extracting" step, and independent claim 56 is also similar. In contrast, Wahbe simply detects the existence of a potentially dangerous write instruction in a group, conditional or not.

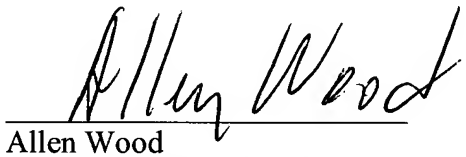
Since the remaining claims depend from the independent claims discussed above and recite additional limitations to further define the invention, they are patentable along with their independent claims and need not be further discussed. It is nevertheless noted that new dependent claims 67-72 have been added to specify that 'the initial conditional breakpoint occurs at a program statement in the loop that has been designated by an operator of the program debugger." In contrast, Wahbe's scheme automatically detects potentially troublesome write instructions in a loop (or outside a loop, for that matter),

rather than examining program statements of particular concern to an operator who is debugging a program.

It is noted that this application has been amended to include six additional dependent claims. Accordingly, an additional claim fee of $108 should be charged to the deposit account of the Assignee of this application.

For the foregoing reasons, it is respectfully submitted that this application is now in condition for allowance. Reconsideration of the application is therefore respectfully requested.

Respectfully submitted,



Allen Wood
Registration No. 28,134
Customer No. 23995
(202) 326-0222
(202) 408-0924 (facsimile)

AW:rw